CLAIMS

1. A method of validating a syntactical statement employing a stored syntax tree representing all possible syntax options by means of a network of junction nodes and data nodes between a root node and an end node, such that all paths through the tree lead to the end node, said method comprising the steps of:

passing said syntactical statement to the root node and parsing said syntactical statement into elementary tokens in the root node;

maintaining the location of a current node in the syntax tree, whereby said current node is initially the root node;

returning potential nodes that can be selected from the current node and their distances from the current node;

in response to said returning step, comparing the potential nodes to the stored tokens and selecting a potential node corresponding to one of said stored tokens if such a corresponding node exists;

updating the location of the current node to that of a selected node,

repeating said returning, comparing and selecting steps wherein the syntactical statement is validated if the end node is reached.

2. A method as claimed in claim 1, further comprising the step of:

creating a table to store said tokens and an entry representing said end node and marking said tokens as "found" in response to said selection of a potential node.

3. A method as claimed in claim 2, comprising the further step of:

initially marking said tokens and end node entry in the table as "not found".

4. A method as claimed in claim 1, in which the distance between a potential node and said current node is measured by enumerating the number of nodes between said potential node and said current node.

5. A method as claimed in claim 2, in which said step of selecting a potential node further comprises the steps of:

verifying successfully if said potential node corresponds to a stored token;

verifying successfully if said potential node is closest in distance to said current node compared with the remaining potential nodes, and

marking said stored token as "found" in the table.

6. A method as claimed in claim 1, in which said syntax tree comprises branched nodes, whereby said branched nodes represent optional tokens or a start node of a sub-tree.

7. A method as claimed in claim 6, in which if a branched node represents a start node of a sub-tree, said sub-tree comprises further junction nodes and/or data nodes.

8. A method as claimed in claim 7, in which sub-trees are nested hierarchically if a sub-tree comprises at least one further start node of a sub-tree.

9. A method as claimed in claim 6, in which said comparing step further includes the step of:

verifying if a potential node is a start node of a sub-tree.

10. A method as claimed in claim 1, in which said junction nodes are linked to any number of junction nodes or data nodes and in which said data nodes are linked to a single junction node.

11. A method as claimed in claim 1, in which said syntactical statement comprises a textual string.

12. A system for validating a syntactical statement comprising:

a stored syntax tree representing all possible syntax options by means of a network of junction nodes and data nodes between a root node and an end node, such that all paths through the tree lead to the end node, whereby said syntactical statement is initially passed to the root node;

means for parsing said syntactical statement into elementary tokens in the root node;

a table to store the tokens, and entries representing the end node of the syntactical statement;

means for maintaining the location of a current node in the syntax tree, whereby said current node is initially the root node;

means for returning potential nodes that can be selected from the current node and their distances from the current node;

means for comparing the potential nodes to the stored tokens and means for selecting a potential node of said potential nodes corresponding to one of said stored tokens if such a corresponding node exists; and

means for updating the location of the current node;

whereby said syntactical statement is valid if said end node is reached.

13. A system as claimed in claim 12, in which said system further comprises:

means for marking said tokens as "found" in response to selection of a potential node by said selecting means.

14. A system as claimed in claim 12 further including

means for initially marking said tokens and end node entries in the table as "not found".

15. A system as claimed in claim 12, further comprising:

means for enumerating the number of nodes between said potential node and said current node, in order to provide the distance between a potential node and said current node.

16. A system as claimed in claim 12, in which said means for selecting a potential node further comprises:

first means for verifying if said potential node corresponds to a stored token; and

second means for verifying if said potential node is closest in distance to said current node compared with the remaining potential nodes.

17. A system as claimed in claim 12, in which said system further comprises:

if the end node is reached, means for confirming the syntactical statement is valid if all stored tokens, including the end node entry in the table are marked as "found".

18. A system as claimed in claim 12, in which said syntax tree further comprises:

branched nodes, whereby said branched nodes represent optional tokens or a start node of a sub-tree.

19. A system as claimed in claim 18, in which if a branched node represents a start node of a sub-tree, said sub-tree comprises:

further junction nodes and/or data nodes.

20. A system as claimed in claim 19, in which said system further comprises:

means for nesting sub-trees hierarchically if a sub-tree comprises at least one further start node of a sub-tree.

21. A system as claimed in claim 20, in which said comparing means further comprises:

means for verifying successfully if a potential node is a start node of a sub-tree.

22. A system as claimed in claim 12, in which said system further comprises:

junction nodes that are linked to any number of junction nodes or data nodes and data nodes that are linked to a single junction node.

23. A system as claimed in claim 12, in which said syntactical statement is a textual string.

24. A syntax checker comprising:

a stored syntax tree representing a body of valid syntax; and

a table for holding elementary tokens of syntax making up a syntactical statement to be checked;

said syntax tree comprising root and end node objects joined by a network of junction node objects and data node objects, said data node objects representing options in the syntax including tokens, such that each junction node object may link to an unlimited number of other junction node objects and data node objects, and each data node object only links to a singular junction node object, whereby all pathways through the network eventually terminate in said end node object;

each of said junction node objects, being effective to evaluate linked data node options following said junction node object so that any tokens in said table

corresponding to a linked data node object are marked as "found",

whereby said syntactical statement is progressively compared with said syntax tree either until said end node is reached, indicating the syntax of the statement is valid, or until a corresponding data object is not found, indicating said syntax is not valid.

25. A syntax checker as claimed in claim 24 wherein said syntax tree further comprises a sub-tree having a start node represented by a corresponding syntax option in a data node.

26. A syntax checker as claimed in claim 24 wherein said syntax tree comprises a plurality of sub-trees, nested hierarchically, each nested sub-tree having a respective start node represented by a corresponding syntax option in a data node.

27. A computer program recorded on a medium consisting of instructions which, when executed on a computer, perform a method of validating a syntactical statement employing a stored syntax tree representing all possible syntax options by means of a network of junction nodes and data nodes between a root node and an end node, such that all paths through the tree lead to the end node, said method comprising the steps of:

passing said syntactical statement to the root node and parsing said syntactical statement into elementary tokens in the root node;

maintaining the location of a current node in the syntax tree, whereby said current node is initially the root node;

returning potential nodes that can be selected from the current node and their distances from the current node;

in response to said returning step, comparing the potential nodes to the stored tokens and selecting a potential node corresponding to one of said stored tokens if such a corresponding node exists;

updating the location of the current node to that of a selected node,

repeating said returning, comparing and selecting steps wherein the syntactical statement is validated if the end node is reached.

28. A computer program as claimed in claim 27, in which said method further comprises the step of:

creating a table to sotre said tokens and an entry representing said end node and marking said tokens as "found" in response to said selection of a potential node.

29. A computer program as claimed in claim 28, comprising the further method step of:

initially marking said tokens and end node entry in the table as "not found".

30. A computer program as claimed in claim 27, in which the distance between a potential node and said current node is measured by enumerating the number of nodes between said potential node and said current node.

31. A computer program as claimed in claim 27, in which said step of selecting a potential node further comprises the steps of:

verifying successfully if said potential node corresponds to a stored token;

verifying successfully if said potential node is closest in distance to said current node compared with the remaining potential nodes, and

marking said stored token as "found" in the table.

32. A computer program as claimed in claim 27, in which said syntax tree comprises branched nodes, whereby said branched nodes represent optional tokens or a start node of a sub-tree.

33. A computer program as claimed in claim 32, in which if a branched node represents a start node of a sub-tree, said sub-tree comprises further junction nodes and/or data nodes.

34. A computer program as claimed in claim 32, in which sub-trees are nested hierarchically if a sub-tree comprises at least one further start node of a sub-tree.

35. A computer program as claimed in claim 32, in which said method comparing step further includes the step of:

verifying if a potential node is a start node of a sub-tree.

36. A computer program as claimed in claim 27, in which said junction nodes are linked to any number of junction nodes or data nodes and in which said data nodes are linked to a single junction node.

37. A computer program as claimed in claim 27, in which said syntactical statement comprises a textual string.